Negative Authentication and Password Secured Systems

Ali J. Askar, Ammar A. Mahawish, Mohammed Nasser Hussain Al-Turfi

Abstract: ***In nowadays, Software Applications has a significant role in each part of our life, this includes the health system, educational system, the industrial factories and even the nuclear plants. therefore, leaving the use of these Applications kind of an impossible to happen, regardless of the mounting risks that comes through cyber-attacks. In Software Applications the password considers as a gate key to the System, therefore attacking passwords is the first target for any cyber attacker. With all the new famous defenses System, there is no any authority that can set up an international law or a policy to guarantee the implementation of these systems, which lead the users to use a more complicated passwords to secure their systems, which it is not an efficient solution. In this Paper, the status of password systems will be presented as well as Implementation of the Negative Authentication.***

Index Terms: Cryptography, Cyber Security, Cipher, Web Applications, Negative Authentication

I. INTRODUCTION

Passwords are used in almost every digital system. Everyone is familiar with logging-in to a PC operating system, accessing their email or signing-in to a social network, but passwords protect access to many other systems like banking terminals and websites, network devices like switches and routers, corporate applications and even entertainment services hooked up to our TVs. Most vendors have been able to implement their own approach to safeguarding those passwords and this has resulted in a mix of safe and insecure systems.

Maintaining password security is a problem of ever-increasing importance. With the increase in online services, the amount of credentials per person has also increased dramatically. News reports of hackers stealing credentials are increasing and so are notices to users requesting password changes. Through this research I'm trying to find how attackers are able to obtain these passwords. The research focuses on the state of the practice and the effectiveness of defense of the password security implementations in preventing the disclosure of the password to an unauthorized party. This does not go into other system vulnerabilities in network, operating system or database.

Revised Manuscript Received on June 8, 2019.

Ali J. Askar, College of Engineering, Al-Iraqia University, Baghdad, Iraq, aliasker2@yahoo.com.

Ammar A. Mahawish, College of Engineering, Al-Iraqia University, Baghdad, Iraq, ammaralkhafaji@gmail.com

Mohammed Nasser Hussain Al-Turfi, College of Engineering, Al-Iraqia University, Baghdad, Iraq, mohammed_alturfi@yahoo.com.

Although comprehensive security is needed, I'll focus on why it's possible to break the passwords and not on how they gained entrance into the system to get the password data. To properly understand some of these issues I will first describe what password security is.

Afterwards I will continue with a brief history of password security and evolution in recent years. Once the big picture has been established, I will detail the most common components and methods used today in implementing password security. I will go over some of the benefits and disadvantages of each method and how they are linked with each other as a measure taken to try to overcome their major drawbacks. To further strengthen the level of security that the usual methods provide, there have been developments in other approaches to authenticate the users. I will go over some of the most common ones that are already available as well as introduce a new approach called negative authentication.

The third chapter presents a review of policies, regulations and recent security reports by important cyber-security labs to establish the current situation and its impact on the industry. Followed by specific high-profile cases where password security has been compromised and some of the reasons why this happened.

Having reviewed the situation, I will detail the implementation of negative authentication we have been working on. There are different approaches that can be tried and each can have some advantages and disadvantages. This will allow you to better understand its potential advantages.

Finally, I will present my conclusions of the analysis, with some recommendations on how to increase the safety of password security systems. I will review some of its implications and the future research that can be done in this subject.

II. HOW TO SECURE PASSWORD

A. Password Security Definition

Password security is the set of mechanisms or approaches used to protect the characters that compose a password and keep them secret from others. It is of course only one component of overall system security, but it is an essential component." [1]. A good password security system will reliably grant access to a user that provides the correct set of characters and make it very hard for unauthorized users to obtain the original password. To achieve this, the system

architecture involves many layers and components that all contribute to guard the personal information of the user. The whole system needs to handle the following aspects of a user's credentials:

- Length and characters allowed
- Login prompt
- Transmission method
- Validation process
- Secure storage

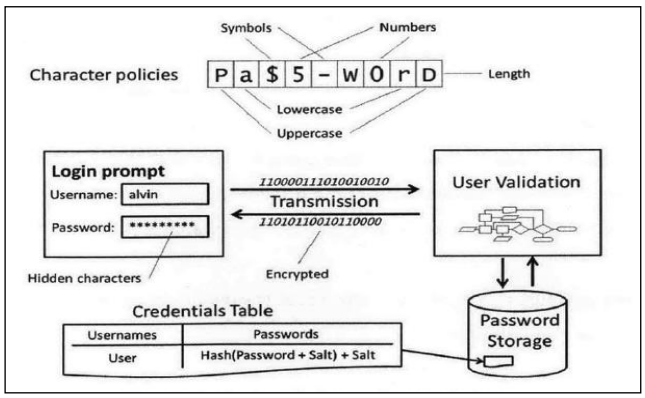

Fig. 1 Architecture of a Password Security System

For this paper the focus is going to be mainly on how the credentials are stored and validated. Weaknesses in any other part of the system present a risk for the system to be compromised allowing unauthorized access or for data to be stolen. Fig. 1 shows a simplification of the different components involved in handling passwords. In this diagram some of the best practices are represented like password character policies, hashing and salting the password. In the following chapters each of the relevant components will be discussed along with some of the benefits and disadvantages of their use and common practices. Fig. 2 shows the User Validation diagram in detail. This has been the basic architecture of password security for many years.

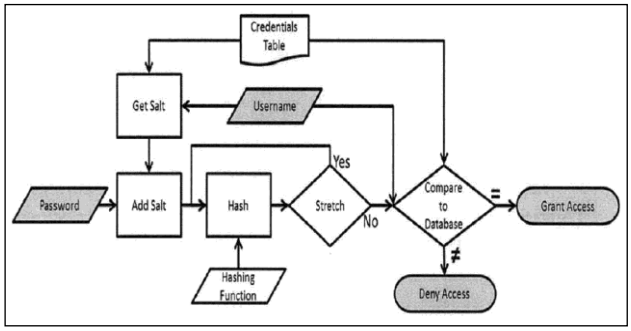

Fig. 2 Flow diagram of a basic Password Security System: User Validation

One of the most important aspects of the storage is the hashing function used. Because this is a complex calculation, this is the main component that determines the time it takes to store and compare passwords. Since there is no way to reverse a password hash to get the original string, an attacker must try many possible combinations with the hope that in time he'll find the correct one. Therefore, the amount of time it takes to hash each attempt becomes the critical factor determining the speed of hacking a password database [1].

B. Password Security Methods

As computer systems became more prevalent, the need to protect passwords became clear. There are a few methods that are commonly used to protect these systems. To properly store a password in a file or database, one or many of these methods are applied to the original password string. The resulting output is what is ultimately written to the storage and to properly authenticate, the system must apply the same methods to a user's credentials when they try to log in. Every one of these approaches may be used in conjunction with others to strengthen the security that they provide, but some of them are not useful if used by themselves.

1) Plain-text databases

A plain-text database is one type of database where the information is stored in a way that can be read back with the original meaning. The main benefit of this approach is that storing passwords in plain-text is the simplest method. It essentially means writing the characters to a file or database. When a user logs in, the password that is entered is compared to the string stored and if they match, the user is allowed access.

2) Hashing

A hash is a one-way function. It is a complex set of operations that regardless of input, produces a fixed length result called a digest. A good one-way function will be practically impossible to reverse, meaning that obtaining the input from the digest cannot be done.

3) Salting

In cryptography a salt represents a set of data added to the message before it is hashed. When applied to passwords, it is generally a set of characters or bits that can be appended or prepended to the password before going through the hashing function. The salt makes the password longer than it originally was and makes the input string to the hashing function different from each other in systems or users that chose the same password. The characters can be static, meaning they are the same for every entry, or random, meaning generated differently by the server for each user. Adding variability to passwords before hashing differentiates the digest and will hide when users share passwords.

4) Stretching

When using hashing algorithms for password protection, there is another issue that decreases the effectiveness of the security. These algorithms were developed to be very fast so that they could also be used to compare larger messages or files [2]. The amount of time it would take to calculate a hash was measured in milliseconds, therefore thousands of hashes could be calculated per second. This enables someone to check many different combinations of input and compare it to the digest to find passwords [3].

5) Policies

Password policies are restrictions to the length and complexity with which the

users can choose their passwords. A system administrator will implement a set of rules over what users can choose as their password. These rules can work to force the user to increase the length of the password to fulfill at least a set number of characters like a minimum of 8 for example. The policies can also force the user to include numbers, upper case letters or symbols in their password.

C. Security Layers

The password database is not the only layer of security in authentication systems. Every implementation is slightly different, but all of them share layers of security. The whole system is a group of components that work together to provide services to the users. The database has its authentication mechanisms to allow or deny access to the data. The portal through which users log in has to process requests and at some point, compare the supplied credentials with what is stored in the database. The operating system in each of the servers involved plays a critical role in running the applications and preventing unauthorized access. The network enables communication between the interfaces of all the components and can also be configured to monitor or block certain communications.

D. Other Methods

With the increased risk of password compromise, many other approaches have been developed and researched to try to provide greater security. Some methods include additional items needed to authenticate, redesigns of the password storage system, extra layers of authentication, among others. Following is a description of some of these methods, but there are many others that can seem weird, strange or novel. There have been proposals about systems where you should pick the color of your password, others where you trust your friends to authenticate you or where you must recognize your friends to get access. We will focus on the ones that are more relevant to this study because of their similarity to my research or for their broad utilization. Each one has its benefits and its weaknesses, and analyzing many of them can help understand how a more robust system can use a combination of approaches.

1) Two-factor and Multi-factor authentication

Multifactor authentication means that the user will need more than one factor to authenticate. There are three categories of authentication factors: something we know, something we have or something we are [4]. In recent times we have also seen the rise in the idea there could be a fourth type of factor: something we do [3]. Normally the password is the first factor of authentication that most of us are used to. The password represents a secret that we know, and like this there are many other things that fall into this category. For example, the questions we get asked when we forget a password: "What was your first pet's name?", "Where did you grow up?", etc.

2) Threshold Cryptography

Current authentication systems like the ones we have been looking at throughout this paper are based on one database containing all the data necessary to authenticate a user. Another approach is to split the data between servers so that authentication requires multiple parts of a key to get access [5]. The user does not need to change anything, as far as they are concerned, they are inputting their username and password the same way they have always done. Behind the curtain, in the datacenter, the credentials are treated differently. This system uses complex math so that the data in only one of the servers is useless, but when enough servers cooperate, they can authenticate the password.

3) Negative Authentication

Up to this point, all the authentication mechanisms are based on storing the credentials of the user and if they match what is being entered to log-in, they are granted access. This is called positive authentication, because it positively identifies the user. Negative authentication takes a different approach to making sure the user is who he says he is. Negative authentication as proposed by Dr. Dasgupta [6] takes the credentials that are sent by the user trying to log in and compares them to everything that it knows is not a valid username and password combination.

III. SECURITY REPORTS

The past few years have been eventful in data breaches where password information has been compromised. In here I present relevant findings in several security reports by done by major security labs. All of these reports gather information from multiple sources and then sort and quantify real world issues that happened during the period of the report. All of them apply to all or part of 2012 and are the most recent reports available from each of the organizations.

The main findings reflect some big issues with password security and the repercussions that this has on further attacks. Companies are disclosing more incidents where data was stolen. The misuse of passwords ranks very high as the main vector that hackers use to gain access to a system. From there the next step may be financial (trying to acquire illicit funds) or gathering data (including sets of passwords that may or may not be hashed.) To get the original passwords to access the system, perpetrators leverage password reuse where employees have their passwords compromised elsewhere and because they use the same password for work, the attacker can log-in with their privileges. Another popular method is with malware designed to steal passwords from the user's computer. Once the user's computer is infected with malware, it can capture information on that system and relay it outwards. Even if the original code for the malware didn't have that function, it usually contacts outside servers to get more commands and install other malicious software.

According to Trend Micro A typical data breach occurs in three phases (Research, Attack and Exfiltrate) [7] as shown in Fig. 3.

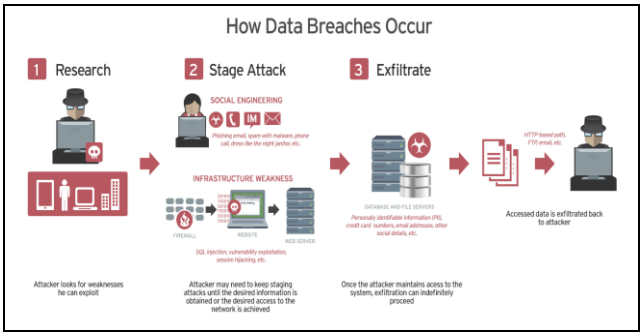

Fig. 3 How data breaches occur (Source: Trend Micro)

IV. NEGATIVE AUTHENTICATION

Negative authentication system also abbreviated NAS, refers to a system of authentication that compares the submitted credentials against a database of values that are known to be invalid combinations of username and password. The total space of valid username and password combinations is very large, and it depends on the method of storage for each particular implementation. For the implementations that we have been working on, the combination of usernames and passwords are stored using 256-bit hashes which means there are 2A256 possible combinations. Every pair of username and password will map onto one point on the negative space, and the hashing algorithms produce a uniform distribution. Beyond this, any set of credentials that's sufficiently large could produce hash collisions.

Fig. 4 represents the possible space of username password combinations and in a) shows a set of 25 pairs of username and password already mapped onto the space, while on b) the resulting negative space. Notice the negative space is not the exact opposite of the positive, but still every valid pair of credentials map to a hole.

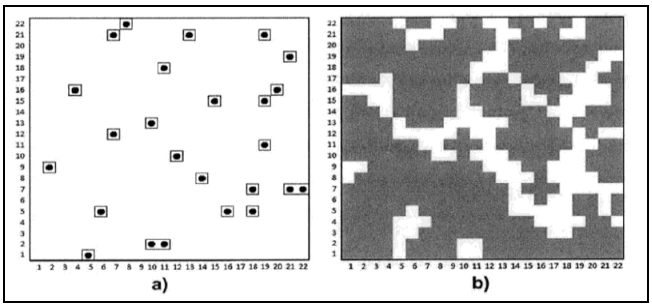

Fig. 4 How Positive and Negative Space (Negative Authentication)

Because the difference between the total possible combinations and the valid set of usernames and passwords is big enough, we can add some uncertainty to the data of negative detectors. The number of valid username and password combinations depends on the amount of users. A system with 10,000 users will have less than 2A10 combinations which would leave 2A256-2A10=2A246 values covered by the negative detectors. In a system with 1,000,000 users we would still have a detector space of at least 2A236. With such a large space, we can have several times the amount of valid credentials as holes or ambiguity in the detectors (also called the confusion parameter). This serves the purpose of hiding the positive database by adding noise to the inverse of the data.

A. Our Approach to Negative Authentication

The implementations that we have worked on stemmed from Ji's and Dasgupta's papers and research [8]. The basis of the system is the sets of valid credentials that are stored in the positive authentication database. From there, the user and password for each account are combined and then hashed and the resulting digest becomes the point in the space that must remain as holes, also named the self-points. One of the advantages of negative authentication is that for it to work hashing the password is mandatory. This will guarantee that at least that level of security is implemented in the system.

Depending on the particular implementation algorithm, the digests go through a varying transformation processes so that they become coordinates in the negative authentication space. Each implementation will have different characteristics for the space and it can be represented in many dimensions. For every dimension, there needs to be a value, therefore the digest is split and each part is assigned to a dimension. Our tests have focused on 2, 4 and N dimensions where N represents the number of bits in the digest. However, the same theory can be applied to any number of dimensions, even 1 dimension.

B. Different Implementations

Structuring data across two dimensions allows for a visually clearer representation of the negative space. This way it can be shown in flat diagrams presented in this thesis. Details on each method of implementation are explained in the next sections. All of these implementations work as Negative Authentication layers that rely on a second layer of positive authentication to authenticate a user with 100% accuracy.

Three parameters affect the accuracy of negative authentication: the area covered by the detectors, the area of the self-points and the resolution of the self-points vs. the actual digest. If the area of the detectors is very close to covering everything but the coordinates resulting from the digest, then it will be easier to inverse the data to get the positive authentication database. If the area of the detectors is small, then it will be very hard to determine the positive database, but the certainty that the users are using the correct credentials to log in is diminished. Each of the implementations offers a varying degree of certainty that can be adjusted by tweaking the aforementioned parameters.

To authenticate users through the negative authentication layer, the supplied credentials are processed in the same manner that was used to generate the self-points for each method. Then the resulting coordinates are checked against the detectors. If there is no match, it means that layer 1 is passed and there is a level of certainty that the credentials are valid. The authentication is then passed to layer 2. In the opposite case, where the point matches a detector, the layer 1 authentication fails and it's known that the supplied credentials are wrong. This

allows the first layer to stop invalid requests from reaching layer 2 and also provides a way of triggering advanced warnings if there are many failed attempts to log-in.

Since negative authentication is still a new design to implement authentication, there is no dominant architecture of how it is built or how it works. The following are examples of approaches that we have been testing so far.

1) Real Space

The Real Space implementation uses hyper spheres in 4 dimensions to represent the areas that are not valid usernames and passwords. These hyper spheres are the detectors and are defined by a point in the 4D space which is the center of the sphere and a radius that is the distance from the center which the detector covers.

To generate the detectors, the process starts with the set of valid usernames and passwords and from there gets the self-points. Every self-point is composed of four segments of equal length. The digest is split into four equal segments that map to the 4 dimensions. From each segment a piece is discarded to add the uncertainty or ambiguity. This is the same thing as lowering the resolution of the space. Each of the self-points then represents a point in a space that is smaller than 2A256.

The next step is to generate the detectors. The space between the self-point is filled with hyper spheres in random coordinates with radii that do not overlap with any self-point. The process continues until a certain coverage is achieved. Then the self-point information is discarded and only the detectors remain as a set of coordinates and radii. Fig. 5 shows the steps in the creation of the detectors. To be able to present it clearly, the diagram shows only 2 dimensions, while the Real Space implementation uses 4. Step a) is the representation of the self-points in the available space. Each one is marked by a point enclosed in a small box. The box represents the minimum resolution that the implementation allows depending on the amount of data that's discarded from the split digest. In step b) the self-points are shown with the hyper sphere detectors and an approximation of the area covered by them in light gray. None of the detectors must cover any self-point. Step c) shows just the detectors remaining as hyper spheres. This resulting database of detectors will be used to authenticate the users in the Negative Authentication level.

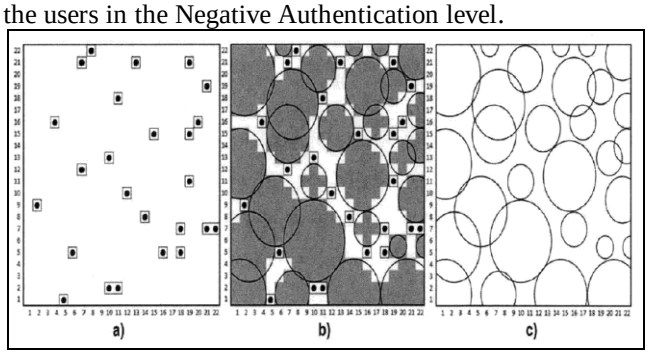

Fig. 5 Real space - Detector Creation (Negative Authentication)

Real Space implementation provides a coordinate space that is complicated to understand at times because it has to be visualized in 4 dimensions. However, computers have no trouble in handling the added complexity. The distance between the points is calculated by getting the Euclidean distance which is the square root of the sum of the distances between the two points squared. This is very similar to calculating the hypotenuse of a triangle, but in 4 dimensions.

One issue with the current implementation is that it takes a long time to generate random points and check them against all the self-points and previous detectors before creating a new detector. As the number of detectors increases, so does the time to check them all.

2) Binary

The Binary implementation is similar to Real Space in that for the detectors it uses the concept of hyper spheres with the coordinates of its center and a radius. However, this implementation goes to N dimensions which is the same as using each bit in the digest as a dimension. If were using a hashing algorithm of 256 bits, the dimension is 256. This allows a simplification on calculating the distance between two points. Since every bit is a dimension, the maximum distance or difference there can be in any dimension between two points is 1. When both points share the same value for a bit 0,0 or 1,1 there is no difference. When they have different values 1,0 or 0,1, the distance is 1. This is called the Hamming Distance. Fig. 6 shows the calculation for two strings. The binary boxes show every bit of the first string stacked on top of the second-string bits. The third row shows a 1 in every column where there's a difference in the strings. When all the 1's in the third row are added, the result is 13 which means we obtain a Hamming Distance of 13. Below the strings is the hexadecimal representation of each which is easier to read than long strings of binary digits.

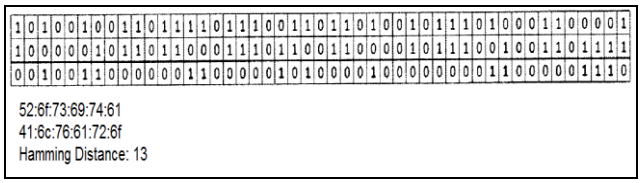

Fig. 6 Binary - Hamming Distance (Negative Authentication)

In Binary space, we don't discard some of the data when generating self-points. Each self-point uses all the bits in the digest. To add the ambiguity or confusion to the set of detectors, the radii of the self-points is increased. The random creation of detectors and their radii makes the self-points very hard to get from only the negative authentication data. The binary implementation shares the issue of slow creation of detectors as the number of these increases because they have to be checked to every previous one created.

3) Grid based

Originally suggested by John Williams, the Grid based implementation splits the available space into a grid [9]. The detectors are the cells that do not contain self-points. Since the cells are part of the total area, this is the same as the reduction of resolution and automatically provides

ambiguity proportional to the size of the grid cell chosen. The self-points only need to map to a grid cell, all the extra data that determines in which position in the cell they are, is discarded. Additional ambiguity can be added by selecting random cells and treating them as if they were also self-points. However, because of the normal distribution of hashing algorithms, choosing random fake cells or increasing the size of the cells produces the same effect. Each time we double the amount of fake cells we could alternatively increase the dimension of the cells by one bit to obtain the same level of ambiguity.

Using a grid pattern in two dimensions allows a much simpler method of finding all the space that is not part of the self-point cells. Therefore, Grid is the fastest method we have tested of creating the detectors. For our implementation, the detectors are rectangles that are one cell high and cover the horizontal area from the beginning of the row to the first cell with a self-point or from the previous cell with a self-point to the next one. Fig. 7 shows the steps in the process. In step a) the self-points are mapped to cells. Step b) introduces random fake cells and then fills up all the remaining space with detectors shown as light gray rectangles. Step c) shows the resulting set of detectors where it's impossible to distinguish the real from the fake self-points. Additionally, every cell that is left blank represents the equivalent of cell-size worth of possible digests that map onto this space. Only one digest is needed to mark a whole cell as self-point, but each cell can contain millions or more possible digests.

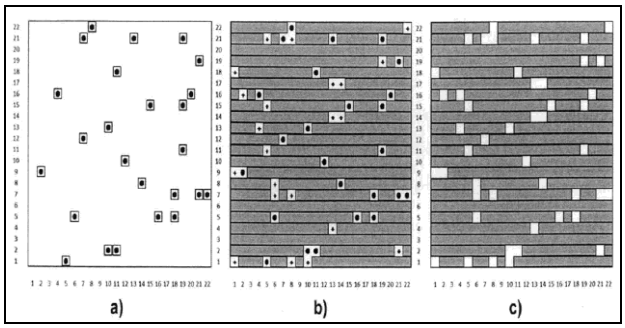

Fig. 7 Grid - Detector Creation (Negative Authentication)

The main advantages of grid space are that it is easy to understand and implement, it is also very fast to create the detectors and therefore can work in almost real-time when there is a user creation or password change.

C. Server Environment and Layers

The different implementation methods describe the behavior of the detectors and the database. A very important part of the system is how it's organized to take advantage of the negative authentication layer. The system is deployed in two layers with different security levels. The first layer is the one that faces the users and is the negative authentication layer. A server receives requests from users to authenticate and dos the initial verification. The second layer contains the positive authentication server that will make sure the user credentials match 100%. In the same security level of layer two, there's an application server in charge of generating the list of detectors and then sending them to the layer one server.

By implementing this layered approach, the communication between the layers may be inspected and reinforced. With the added warning mechanism that negative authentication provides, the whole system can be better equipped to detect irregular behavior that may be a sign of an attack. Knowledge of an attack in progress can help the administrators to deploy countermeasures earlier than without a warning system. This could lead to less instances of compromising the user's passwords. It is important to highlight that the security of the system will depend on its weakest link, and if an attacker were to get access to the layer 2 positive authentication database, it would be no different as it was a traditional one-layer positive authentication system.

NAS has some distinct characteristics like pre-screening requests and getting 99.9% or higher certainty that the user is who they say they are, but without having a database that can be reversed to reveal the user's passwords. Current implementations still have room to grow and mature to a point where systems like this may become more valuable.

V. RESULTS

So far, the research has shown that password security is still a grave issue. There is no indication that the overall safety of the user's credentials is getting better, in fact, as time goes by and computing power increases or becomes cheaper, the situation worsens. The techniques used to optimize password guessing are making brute force attacks much more efficient. Where previously thought that the amount of work needed to crack a password would take years, now takes seconds or minutes.

The sophistication of the tools available to hackers is increasing in every generation; meanwhile the system architecture of a password security system hasn't changed in decades. All of the security enhancements done to password security have been in increasing the length of digests and the time it takes to calculate them. The next generation hashing algorithms are starting to become more common and when well implemented will help keep passwords hidden for another generation. The adaptable stretching of the algorithms may help extend their life, but it requires attention to keep the iteration numbers high enough to represent a decent computational challenge. For this to work, they need to be ubiquitously implemented.

Even with a password system design that has been around for many years, there are constantly public disclosures of large corporations where the best practices were not followed. Current policies do not enforce many of these best practices, but instead just offer them as guidelines or recommendations. The big software vendors are not doing a good enough job to make sure they keep up to date with password security implementations and promote them with their customers. For many system administrators this represents a confusing landscape and creating a lack of knowledge on how to implement their system.

New system designs like negative authentication and others, offer only slightly better security than the systems that are implemented with the current design to the best of its capabilities. However, the new approaches force the use of hashes and can take the opportunity to define their system as also requiring salting and stretching. This clean slate approach may be the largest short-run benefit to the overall system security.

Negative authentication by itself in the implementations we have seen so far is not going to solve the big problems with password security that we have now. Going forward, It certainly provides an interesting vector to search for new approaches to keeping passwords safe. The change in design of password security systems can benefit from the advances we have made up until now by using the same components (hashing, salting, stretching) but changing the way they work with each other and how users are validated. Other components like multifactor authentication can be more integrally tied into the system to add versatility and thoroughness. A new security system design that proves to be safer than the current one will allow us to step away from just making more complex hashing algorithms.

Although we can keep increasing the computational expense of calculating hashes almost without limit, the human aspect of choosing passwords does have its limits. Every day there are more services online that require passwords, and as password become longer, they are harder to remember. Users are now also forced to follow character policies that may not have been well thought out. The combination of all these factors makes password reuse more tempting and, in many cases, users rely on weak passwords. We have seen how one compromised password that is reused between personal sites and work access, can lead into a serious security breach.

VI. CONCLUSION AND RECOMMENDATIONS

A. Recommendations

Institutional measures need to be taken to ensure systems administrators implement defenses for all known exploits. Currently, the gap between solution and vulnerability is too large. It is important to recognize security requires constant vigilance. Security vulnerabilities, like technology, are constantly changing. Security needs to be engaged in a continuous cycle of self-improvement.

Unfortunately, as has been discussed, many organizations fail to implement even the most basic measures. Today, there should be no password security system that does not take advantage of hashing, salting and stretching to protect its passwords.

Companies and users alike should push software vendors to provide easier to implement security in their systems. Most administrators will have enough work to worry about and they want their ROI to come as soon as possible. Vendors and providers should make their software easy to use and offer a clear way of managing password security implementations and updates to it.

Users should hold companies accountable for data breaches, and push for more stringent laws and regulations. They should also insist that companies implement security measures that conform to current best practices. In this age of social networks, if online service providers feel the pressure from its users, they'll be more inclined to implement secure policies, even when it's difficult to verify that they are actually doing it.

B. Implications

Password security has been a concern for many years, but the mechanisms to protect them have not changed very much. The techniques used to secure password databases have been around since the 70's. Through time we have seen the complexity of the algorithms increase to cope with advances in hardware speed and optimizations in the approaches used to calculate hashes. The fact remains in that in the best implemented cases we are still using almost the same security that was designed over 35 years ago.

Many companies are not implementing best practices in password security. Even with such old concepts we find that large corporations have trouble implementing all security measures correctly. Part of the corporation may have recently faced a cyber-attack, but that doesn't immediately cause all the organization to revise their systems and fix them right away.

Other than ignorance and confusion, there are no reasons why large systems can't implement best practices in keeping their passwords secure. When users start pushing companies for accountability and regulators to police the environment, we'll finally see changes that impact their bottom line and consequently drive improvements.

As with any complex system, especially where humans are involved, all developments need to take into account all the components and realize that the weakest one might compromise the whole system.

C. Remaining Questions and Future Work

There are still many approaches that can be taken to design new systems to handle password security. In particular negative authentication could provide a solution that is good enough to authenticate users and does not provide the data needed to trace back to the original passwords. Two approaches for future research come to mind. The first: use only Negative Authentication without Positive Authentication, and the second to link the two layers of negative and positive in a way that obtaining only one of the parts is not enough to work out the passwords.

- Where negative authentication becomes the only layer in which passwords are involved, this would not guarantee 100% accuracy, but it would be very close. Augmented with other systems in multifactor authentication can provide the added certainty required for a functional deployment.
- When linking both layers, the system should need something obtained from successfully traversing layer 1 for it to be able to process layer 2. Currently the easiest

component to manipulate in such a way may be the salt.

Apart from the required technical aspects, a more detailed study into why system administrators don't know the appropriate security measures may help determine the root causes. From there the best course of action to instruct them in the future can be developed. In the hope that there may come a time when everyone involved understands enough to make a safe system.

ACKNOWLEDGMENT

I am indebted to my mother, my late father, and my wife for their continuous love and patience. None of this would have been possible without their encouragement and support.

REFERENCES

[1] Morris, R., & Thompson, K. (1979). Password Security:A Case History. Communications of the Association for Computing Machinery. Murray Hill, NJ: ACM.

[2] Nielsen, P. M. (2012, June 6). Storing Passwords Securely. http://throwingfire.com/storing-passwords-securely

[3] Peslyak, A., & Marechal, S. (2012, December). Password security: past, present, future. Oslo, Norway. http://www.openwall.com/presentations/Passwordsl2-The-Future-Of-Hashing/

[4] Burr, W. E., Dodson, D. F., Newton, E. M., PerIner, R. A., Polk, W. T., Gupta, S., & Nabbus, E. A. (2011). NIST Special Publication 800-63-1 Electronic Authentication Guideline. NIST - National Institute of Standards and Technology, Computer Security Division, Gaithersburg, MD.

[5] http://csrc.nist.gov/publications/PubsSPs.html

[6] Simonite, T. (2012, October 9). To Keep Passwords Safefrom Hackers, Just Break Them into Bits. MIT Technology Review:

[7] http://www.technologyreview.com/news/429498/to-keep-passwords-safe-fromhackers-just-break-them-into-bits

[8] Dasgupta, D., & Azeem, R. (2007, April 15). An Investigation of Negative Authentication Systems. http://seciab1.cs.memphis.edu/files/papers/Antipion.pdf

[9] 19. Data Breach, How data breach occur, Trend Micro:

[10] https://www.trendmicro.com/vinfo/gb/security/definition/data-breach

[11] i, Z., & Dasgupta, D. (2004). Real-Valued Negative Selection Algorithm with Variable-Sized Detectors. http://www.zhouji.net/prof/438reformat.pdf

[12] Williams, J. R. (2013). Professor of Information Engineering, Deptartments of Civil and Environmental Engineering and Engineering Systems Division at Massachusetts Institute of Technology.

AUTHORS PROFILE

Ali J. Askar Al-Khafaji is currently working at College of Engineering/Al-Iraqia University, as a Lecturer as well as Head of Registration Office. Experienced in Data Mining, Cyber Security & Web Development from 2008 until now. Currently holds a degree of Master of Computer Science from Pune University, India. Research interest includes Cloud Computing, Data Mining and Cyber Security. Having three Publications in Several Journals. Completed more than sixteen projects and has a vast experience in the field of Computer Programming.